

The Open Grid Services Architecture

Kate Keahey

keahey@mcs.anl.gov

Ian Foster

foster@mcs.anl.gov

Overview

- Introducing the main players
 - ➔ ♦ **Grid Computing**
 - ♦ Globus Toolkit
 - ♦ Web Service (example)
- The Shape of an OGSA Grid
- The Open Grid Services Architecture
 - ♦ Grid Services: what are they? (example)
 - ♦ WSDL conventions and extensions
 - ♦ OGSA interfaces and behaviors (examples)
 - ♦ OGSA Security
- OGSA: status and future

Requirements Include ...

- Online negotiation of access to services and resources: who, what, why, when, how
- Establishment of applications and systems able to deliver multiple qualities of service
- Other:
 - ◆ Dynamic formation and management of Virtual Organizations (VOs)
 - ◆ Autonomic management of infrastructure elements
- In short: open, extensible, evolvable infrastructure

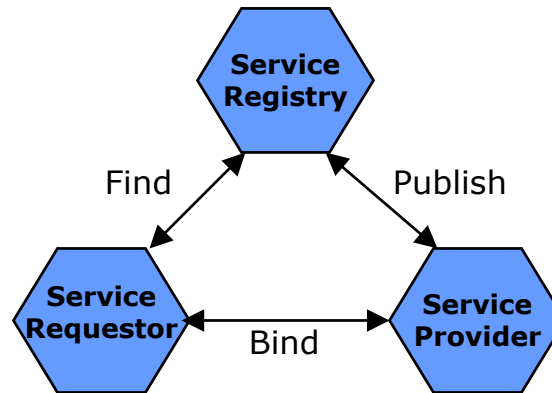
Overview

- Introducing the main players
 - ◆ Grid Computing
 - ◆ Globus Toolkit
 - ➔ ◆ **Web Services (example)**
- The Shape of an OGSA Grid
- The Open Grid Services Architecture
 - ◆ Grid Services: what are they? (example)
 - ◆ WSDL conventions and extensions
 - ◆ OGSA interfaces and behaviors (examples)
 - ◆ OGSA Security
- OGSA: status and future

What is a Web service?

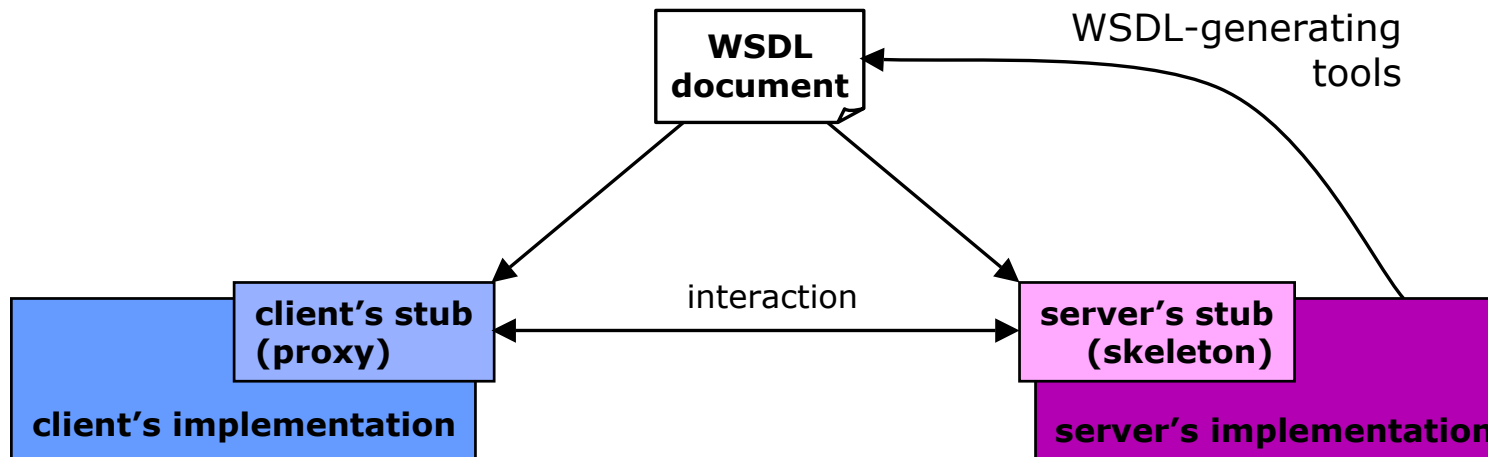
- Web service is an entity that can be:
 - ◆ Described (using WSDL)
 - ◆ Published
 - ◆ Discovered
 - ◆ Invoked by a client
- W3C technology standardization process
- Often associated with specific technologies and implementations
 - ◆ Standards: XML, WSDL, SOAP, UDDI
 - ◆ Implementations: WebSphere, .NET, others...

Service-Oriented Architecture



- **Publish**
 - ◆ WSDL: Web Services Description Language
 - ◆ UDDI: Universal Description, Discovery & Integration
- **Find**
 - ◆ WS-Inspection
- **Bind**
 - ◆ SOAP: Simple Object Access Protocol

WS: Mode of Operation



- **Stubs:**
 - ◆ Serialize/deserialize (encoding)
 - ◆ Implement interaction
- **WSDL-generating tools**
 - ◆ Significantly facilitate working with Web services
 - ◆ Strive to make the process transparent

WSDL Document Structure

- WSDL: Web Services Definition Language
- Document structure:
 - ◆ Service Description
 - ◆ Implementation Details
- Service Description
 - ◆ Elements
 - PortType (~ class)
 - Operations (~ method)
 - Messages, message parts (~ parameters)
 - Types (type definitions)
 - ◆ Used for
 - Generating stubs and skeletons
 - Service discovery

WSDL Document Structure (cntd)

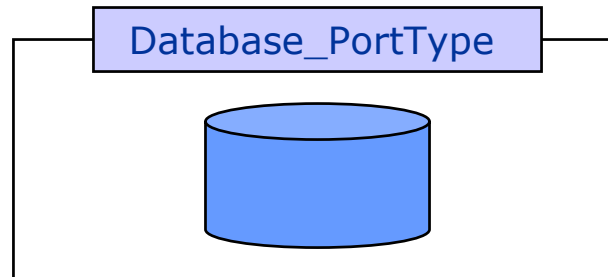
- Implementation Details
 - ◆ Binding
 - Messaging protocol (eg. SOAP)
 - Message Interpretation (eg. RPC or literal)
 - Data-encoding model (eg. SOAP or literal encoding)
 - Transport protocol (eg. HTTP or FTP)
 - ◆ Port: describes service endpoint
 - ◆ Service Element: groups Port elements together
- Others:
 - ◆ Definition: root element of a SOAP document

Web Service Technologies

- Simple Object Access Protocol (SOAP)
 - ◆ XML-based messaging protocol
 - ◆ Independent of the underlying transport protocol
 - HTTP, FTP, etc.
- WS-Inspection
 - ◆ XML language and conventions for locating service descriptions
 - ◆ WSIL: WS Inspection Language
 - ◆ Service description
 - Link to WSDL document
 - UDDI entry
- Other
 - ◆ WSFL: Web Services Flow Language

WS Example: Database Service

- WSDL definition for “Database_PortType” defines operations and bindings, e.g.:
 - ◆ QueryOperation(Query, Response)
 - ◆ Accessible over SOAP



Database: Service Description

```
<types>
  <schema targetNamespace="http://samples.ogsa.globus.org/database/database.xsd"
    xmlns="http://www.w3.org/2001/XMLSchema">

    <complexType name="query">
      <sequence>
        <element name="send_query" type="string"/>
      </sequence>
    </complexType>
  </schema>
</types>

<message name="myDatabaseQuery">
  <part name="query_parameter" type="query"/>
</message>

<message name="myDatabaseResponse">
  <part name="response_parameter" type="string"/>
</message>

<portType name="Database_PortType">
  <operation name="databaseQueryOperation">
    <input message="tns:myDatabaseQuery"/>
    <output message="tns:myDatabaseResponse"/>
  </operation>
</portType>
```

"parameter" type

"parameter"

"class"

"method"

Database: Implementation

use SOAP

interpret as RPC call

use http for transport

```
<binding name="Database_Binding" type="tns:Database_PortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="databaseQueryOperation">
    <soap:operation soapAction="do_databaseQueryOperation"/>
    <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        use="encoded" namespace="http://samples.ogsa.globus.org/database"/>
    </input>
    <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        use="encoded" namespace="http://samples.ogsa.globus.org/database"/>
    </output>
  </operation>
</binding>

<service name="Database_Service">
  <port name="Database_Port" binding="tns:Database_Binding">
    <soap:address location="http://ept.mcs.anl.edu:8080/axis/services/Database_Port"/>
  </port>
</service>
```

use SOAP encoding

the service is located here

Web Services Evaluation (+)

- Key to success:
 - ◆ Emphasize protocols rather than APIs
 - ◆ Build on established technologies and protocols
 - ◆ Web-wide rather than enterprise-wide scope
 - ◆ A set of independent technologies
 - ◆ Industry support

Web Services Evaluation (-)

- Developing technology:
 - ◆ Lack of standard language bindings
 - ◆ Others
- Web Services applied to Grids:
 - ◆ WS describe persistent services
 - For Grids we must also support transient instances
 - Lifecycle management issues
 - ◆ Need to provide information about a service
 - Need ways to access that information
 - ◆ Implications on how services are managed

Overview

- Introducing the main players
 - ◆ Grid Computing
 - ◆ Globus Toolkit
 - ◆ Web Service (example)
- ➔ • The Shape of an OGSA Grid
- The Open Grid Services Architecture
 - ◆ Grid Services: what are they? (example)
 - ◆ WSDL conventions and extensions
 - ◆ OGSA interfaces and behaviors (examples)
 - ◆ OGSA Security
- OGSA: status and future

WS+Grids: Benefits of the Union

- Service orientation
 - ◆ virtualize resources
 - ◆ unify resources/services/information
- Capitalize on useful WS properties
 - ◆ Standards for service description and discovery
 - ◆ Leverage commercial efforts
- Refactor Globus protocol suite to enable common base and expose key capabilities
- Provide a unifying architecture for computational Grids

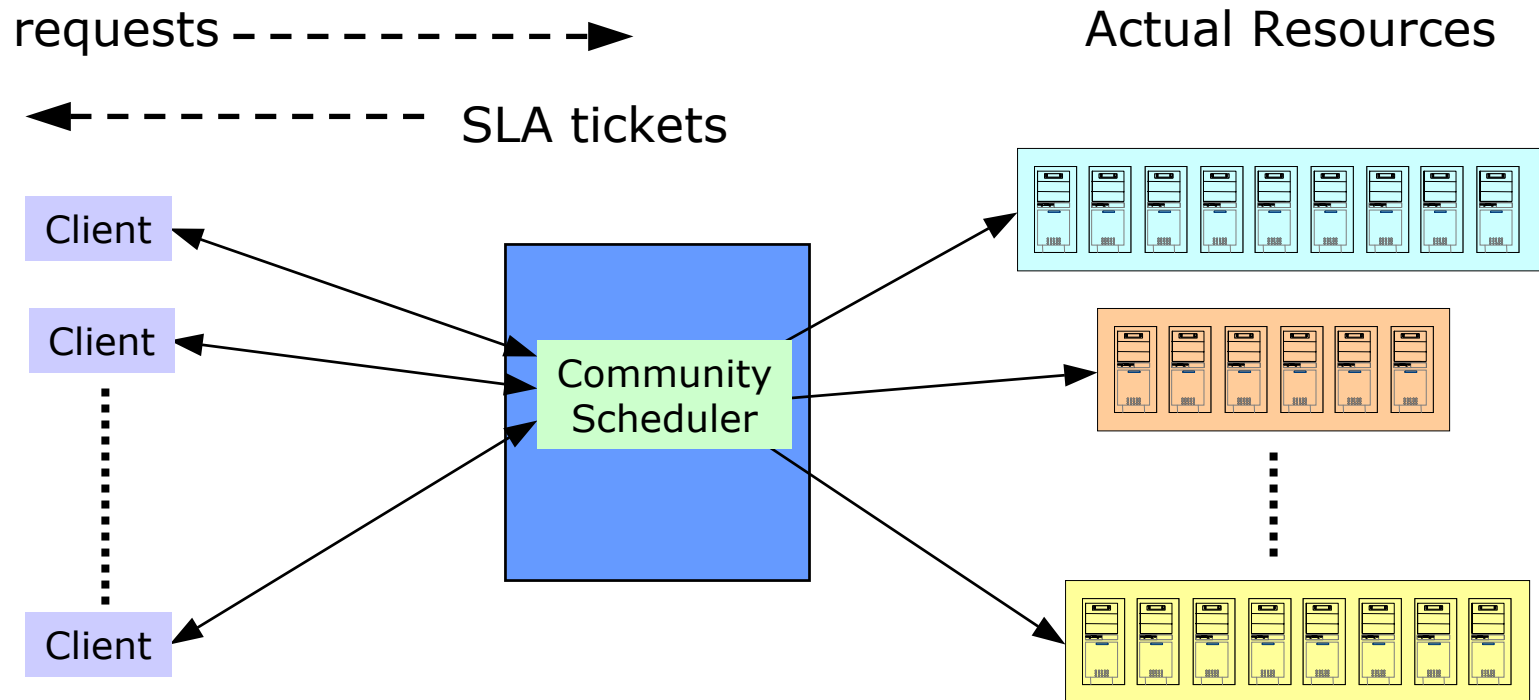
Globus Toolkit Refactoring

- Grid Security Infrastructure (GSI)
 - ◆ Used in Grid service network protocol bindings
 - ◆ Also: Security Services
- Meta Directory Service 2 (MDS-2)
 - ◆ Native part of each Grid service:
 - Discovery, Notification, Registry, RegistryManagement
- Grid Resource Allocation & Mngt (GRAM)
 - ◆ Job Manager Service
 - ◆ Gatekeeper -> Factory for job mgr instances
- GridFTP
 - ◆ Refactor control channel protocol
- Other services refactored to use Grid Services

Moving Forward with Grid Services

- Benefits of service orientation
 - ◆ Focus on interface
 - Minimal shared understanding between interacting entities
 - ◆ Local/remote transparency
 - ◆ Modularity, Reusability, etc.
- Virtualization
 - ◆ Encapsulation of diverse implementation behind a common interface
 - ◆ Defining interactions with services in terms of QoS constraints and Service Level Agreements (SLA)
 - ◆ Living up to SLAs: Adaptive behaviors

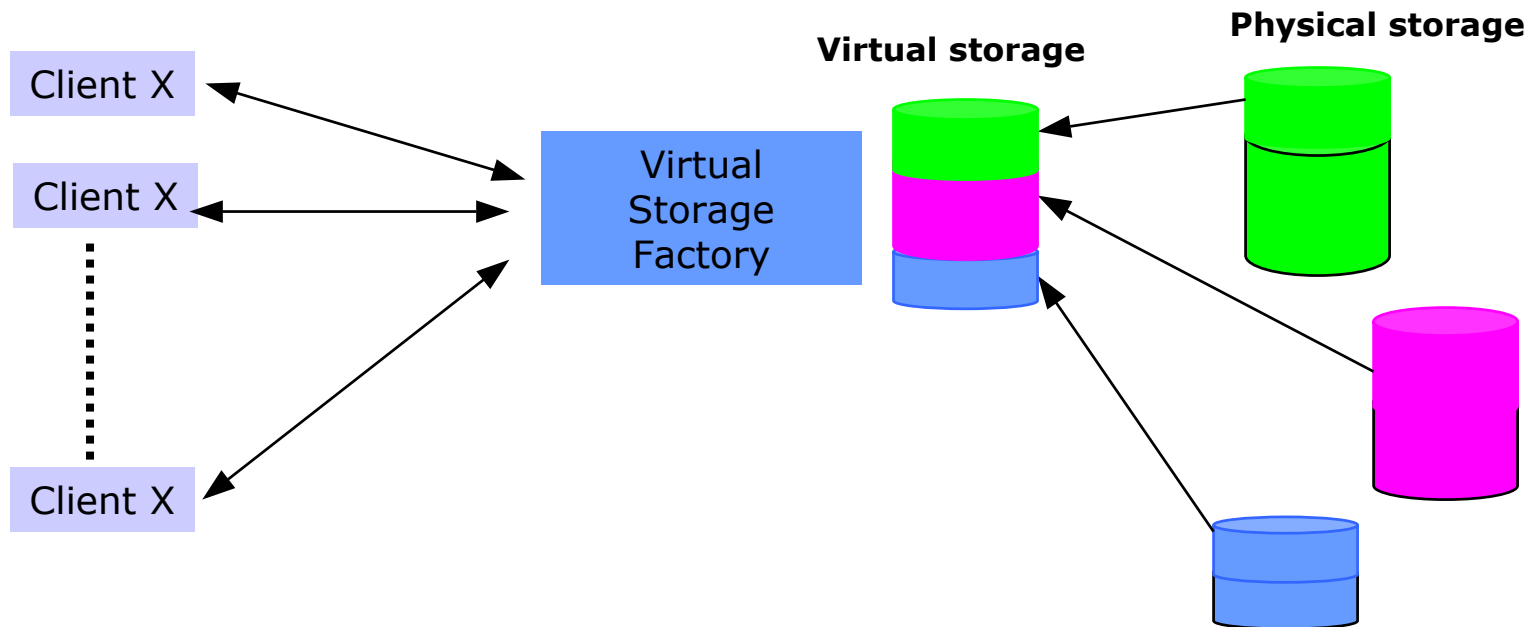
Towards Virtualizing Resources



- SNAP: Service Negotiation and Acquisition Protocol

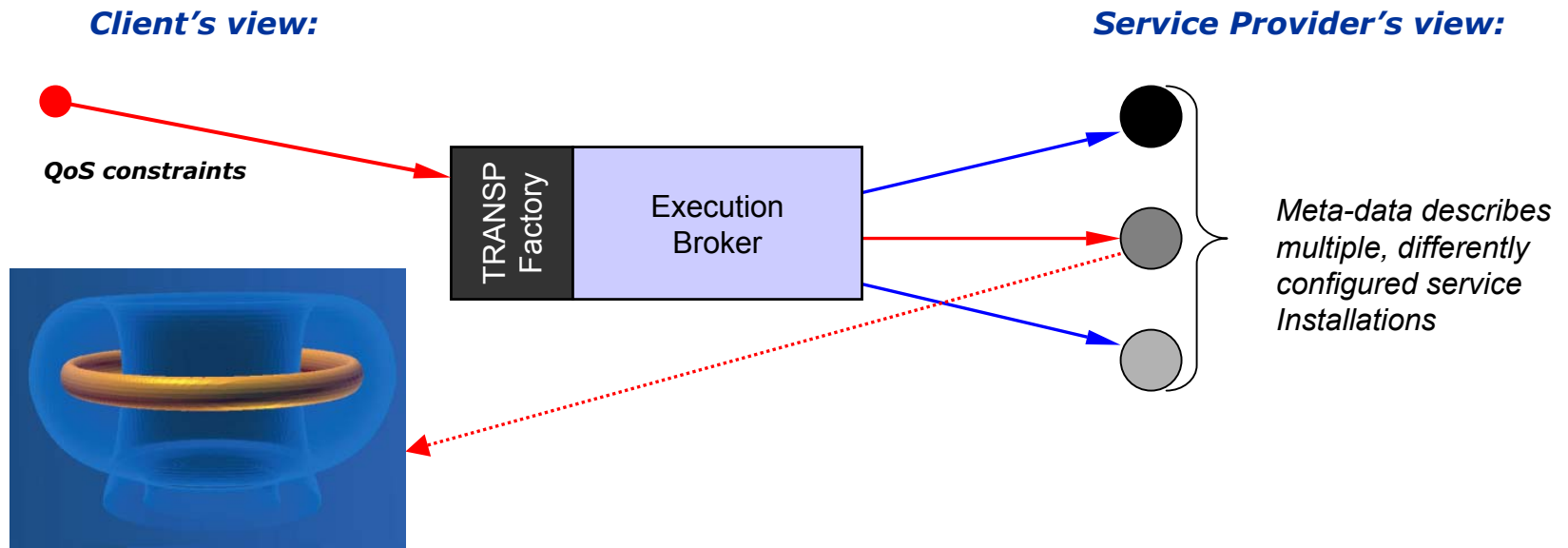
Virtualizing Resources: Example

- Application: Virtual Storage
 - ◆ Garbage collecting unused space in an organization
 - ◆ Providing it to users as “virtual storage”

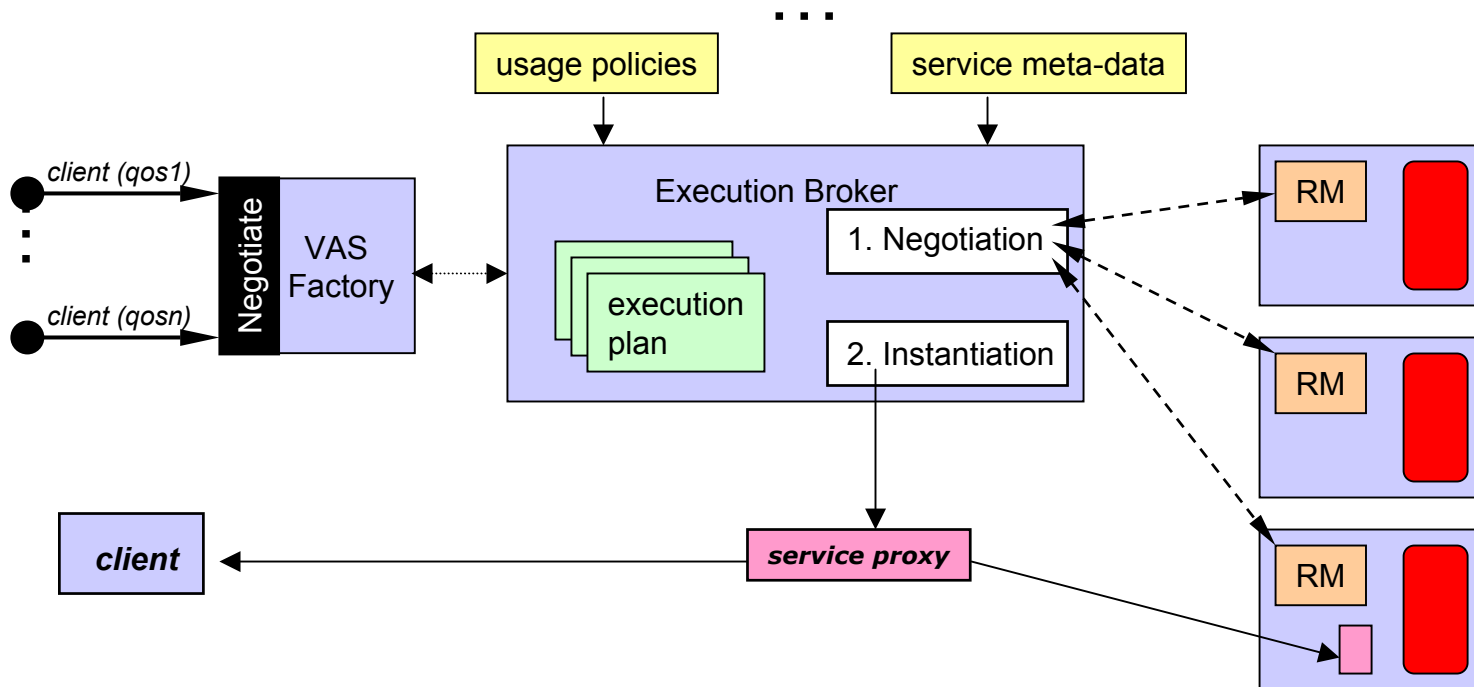


Virtual Application Services (VAS)

- Example: the National Fusion Collaboratory
- Requirements
 - ◆ Codes as “network services” (portability reasons)
 - ◆ Different interaction modes
 - Real-time constraints (betw. Experimental pulses: ~15mins)
 - Batch jobs where accuracy is important



VAS: Behind the scenes



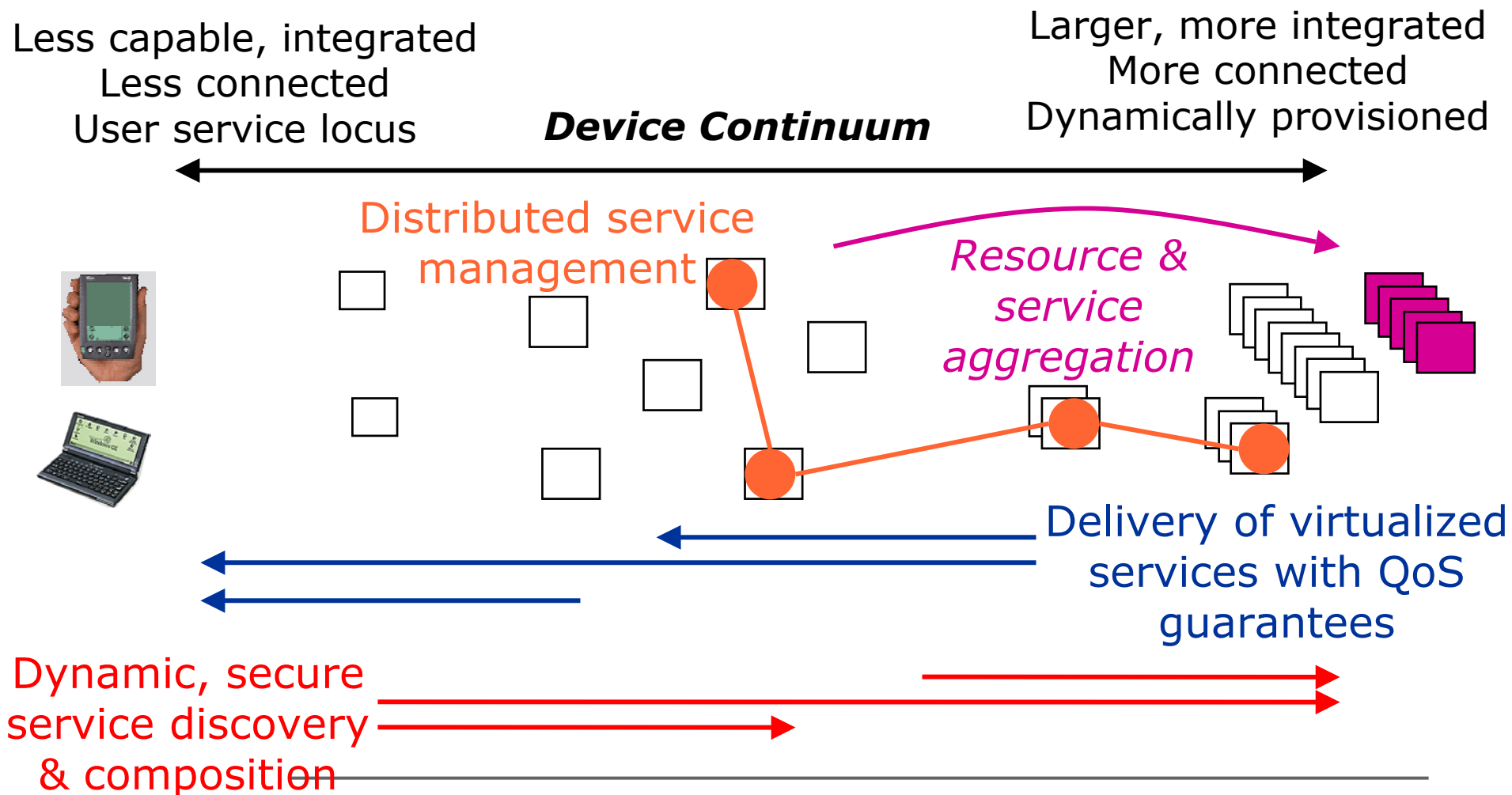
- Adaptive capabilities
- Capable of adjusting to different models

Composing Services

- Resource composition
 - ◆ Complex resource configuration
 - ◆ CPUs, networking, storage...
 - ◆ Redundant configuration to provide for failure
- Application Service Composition
 - ◆ Workflow and orchestration
 - ◆ Constraint-based service discovery
 - ◆ Reliable and Adaptive Workflow execution
 - ◆ Reproducibility
 - Data provenance



Virtualization and Distributed Service Management



Grid Evolution

- Paradigm change:
 - ◆ Spend less time telling the infrastructure *how* to do things
 - ◆ Spend more time telling the infrastructure *what* to do
- Service abstraction
 - ◆ Presents a more intuitive interface to the user
 - ◆ Allows the infrastructure developer to focus on key areas of the infrastructure

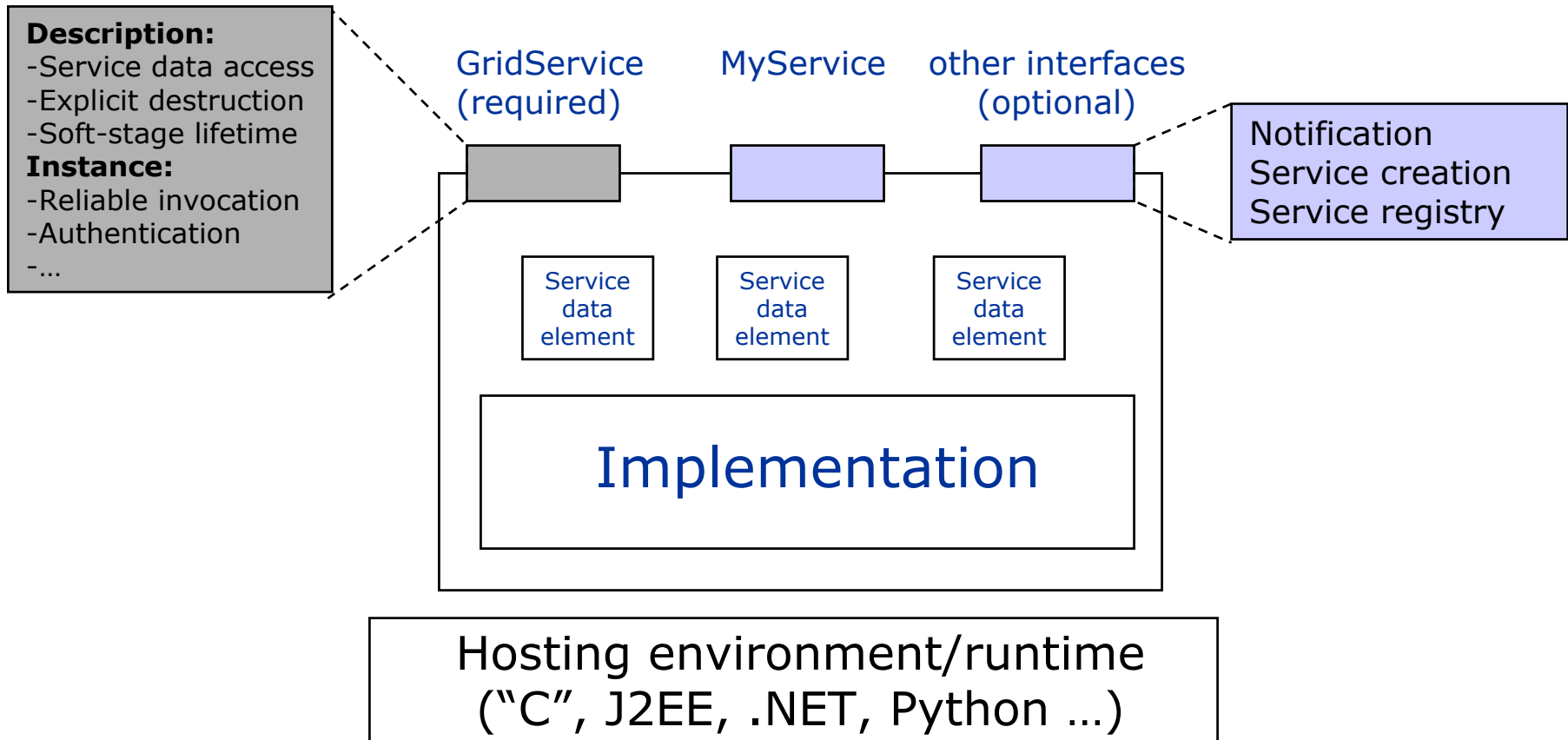
Overview

- Introducing the main players
 - ◆ Grid Computing
 - ◆ Globus Toolkit
 - ◆ Web Service (example)
- The Shape of an OGSA Grid
- The Open Grid Services Architecture
- ➔ ◆ **Grid Services: what are they? (example)**
 - ◆ WSDL conventions and extensions
 - ◆ OGSA interfaces and behaviors (examples)
 - ◆ OGSA Security
- OGSA: status and future

Open Grid Services Architecture

- From Web services
 - ◆ Standard interface definition mechanisms
 - Interface and implementation (multiple protocol bindings)
 - local/remote transparency
 - Language interoperability
 - ◆ A homogenous architecture basis
- From Grids
 - ◆ Service semantics
 - ◆ Lifecycle management
 - ◆ Reliability and security models
 - ◆ Discovery
 - ◆ Other services: resource management, authorization, etc.

The Grid Service



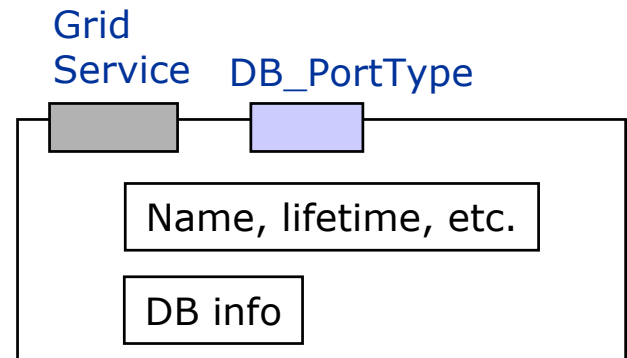
The Grid Service

- A WSDL-defined service that conforms to a set of conventions relating to its interface and behaviors
- Description composed of two parts:
 - ◆ Grid service description
 - Describes how a client can interact with service instances: syntax and semantics (portTypes)
 - Can be used by any number of GS instances
 - ◆ Grid service instance
 - Embodies state
 - Has one or more unique Grid Service Handles
 - Has one or more Grid Service References



Grid Service Example: Database Service

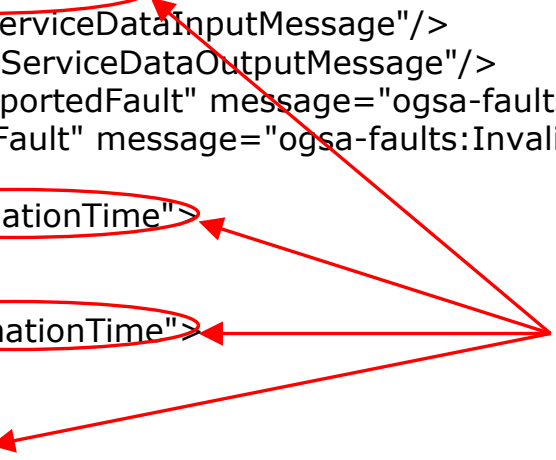
- A DBaccess Grid service will support at least two portTypes
 - ◆ GridService
 - ◆ Database_PortType
- Each has service data
 - ◆ GridService: basic introspection information, lifetime, ...
 - ◆ DB info: database type, query languages supported, current load, ..., ...



The Database Grid Service


```
<portType name="GridServicePortType">
  <operation name="findServiceData">
    <input message="tns:FindServiceDataInputMessage"/>
    <output message="tns:FindServiceDataOutputMessage"/>
    <fault name="QueryNotSupportedFault" message="ogsa-faults:QueryNotSupportedFault"/>
    <fault name="InvalidQueryFault" message="ogsa-faults:InvalidQueryFault"/>
  </operation>
  <operation name="setTerminationTime">
    ...
  </operation>
  <operation name="getTerminationTime">
    ...
  </operation>
  <operation name="destroy">
    ...
  </operation>
</portType>
```

Grid Service Functionality



```
<portType name="Database_PortType" extends="gsdl:GridService">
  <operation name="databaseQueryOperation">
    <input message="tns:myDatabaseQuery"/>
    <output message="tns:myDatabaseResponse"/>
  </operation>
</portType>
```

Database_PortType Inherits from GridService



Open Grid Services Architecture: Fundamental Structure

- 1) WSDL conventions and extensions for describing and structuring services
 - ◆ Useful independent of “Grid” computing
- 2) Standard WSDL interfaces & behaviors for core service activities
 - ◆ Necessary for Grid computing
- 3) Higher-level services

Overview

- Introducing the main players
 - ◆ Grid Computing
 - ◆ Globus Toolkit
 - ◆ Web Service (example)
- The Shape of an OGSA Grid
- The Open Grid Services Architecture
 - ◆ Grid Services: what are they? (example)
 - ➔ ◆ **WSDL conventions and extensions**
 - ◆ OGSA interfaces and behaviors (examples)
 - ◆ OGSA Security
- OGSA: status and future

WSDL Extensions and Conventions

- Defined using WSDL extensibility elements
- WSDL conventions and extensions
 - ◆ serviceData: properties of a service that may be queried
 - ◆ serviceDataDescription: formal description of serviceData elements
 - ◆ portType inheritance: recently added to WSDL
 - Extending portTypes
 - ◆ Naming: naming conventions on portType and serviceType
 - ◆ Grid Service Reference (can be a WSDL document)
 - ◆ Grid Service Handle

Service Data

- Describes
 - ◆ Meta-data (info about the service instance)
 - ◆ State data (runtime properties)
- Represented by a Service Data Element (SDE)
 - ◆ Structural
 - Extensibility element in portType
 - Any GS that of this description must implement them
 - ◆ Non-structural
 - Described by serviceDataSet

Service Data Element

- Information:
 - ◆ Name
 - ◆ Type (XML type)
 - ◆ Extensibility attributes
 - Lifetime declarations
 - ◆ goodFrom, goodUntil, availableUntil
 - Application-specific
 - ◆ Extensibility elements
 - Service data value
 - Application-specific

Service Data Descriptions

- Specifies properties (type) of SDEs
- Extends the definitions element
- Interface
 - ◆ Name,
 - ◆ XML type of service data element values conforming to this description
 - ◆ minOccurs, maxOccurs
 - ◆ mutability

Service Data Set

- A set of SDEs
- Each Grid Service must have exactly one service Data Set
- Accessible in two ways:
 - ◆ FindServiceData
 - ◆ Notification
- It must include all structural SDEs
- It may in addition also include some non-structural SDEs

Naming and Change Management

- The change management problem
 - ◆ GS semantics may evolve
 - On the interface level: adding new operations
 - On the implementation level: bug fixes, etc.
 - ◆ Users rely on this behavior
- OGSA requirement: all elements of a GS description must be immutable
 - ◆ Qualified name (namespace and locally unique name) must refer to only one WSDL specification
 - ◆ If a change is needed a new service with a new qualified name must be defined

Naming: Handles and References

- Grid Service Handle (GSH)
 - ◆ Uniquely identifies a service
 - ◆ Has the form of URI
- Grid Service Reference (GSR)
 - ◆ Contains all the information a client needs in order to communicate with a service
 - ◆ Its form depends on the binding
- GSH must be resolved to GSR in order to use a service
 - ◆ Information on how to resolve encoded in the URI
- Separation of name from implementation details facilitates manipulation of a service

Grid Service Handle

- Name in the form of URI
 - ◆ The URI scheme defines the protocol for resolving it
- Properties
 - ◆ GSH is valid for the lifetime of a GS instance
 - ◆ Must not refer to more than one service instance
 - ◆ A GS has at least one GSH
 - ◆ GSH may resolve to different GSRs pointing to the same service
- Resolver protocols
 - ◆ Untrusted (http)
 - ◆ Trusted (https)

Grid Service Reference

- Network-wide pointer to a specific GS instance
 - ◆ Web service binding mechanism
 - ◆ Binding-specific information about the endpoint
 - ◆ May include expiration time (treat is as a hint)
- Binding-specific
 - ◆ SOAP: WSDL document
 - ◆ RMI/IIOP: CORBA-compliant IOR
- May become invalid during the lifetime of an instance (independent lifecycle)
- Many GSRs to a service may exist at the same time
- Use of invalid GSR should result in an exception

Overview

- Introducing the main players
 - ◆ Grid Computing
 - ◆ Globus Toolkit
 - ◆ Web Service (example)
- The Shape of an OGSA Grid
- The Open Grid Services Architecture
 - ◆ Grid Services: what are they? (example)
 - ◆ WSDL conventions and extensions
 - ➔ ◆ OGSA interfaces and behaviors (examples)
 - ◆ OGSA Security
- OGSA: status and future

Standard Interfaces and Behaviors

- Grid Service: basic behavior
- HandleResolver: mapping from GSH to GSR
- Lifecycle
 - ◆ Support transient services
 - ◆ Service instances created by factories
 - ◆ Destroyed explicitly or via soft state
- Notifications
 - ◆ Registering interest and delivering notifications
- Registration
 - ◆ Allows clients to register and unregister registry contents

Grid Service Interface (Recap)

- Must be implemented by all Grid services
- Interface:
 - ◆ FindServiceData
 - Input
 - ◆ QueryExpressionType: query mechanism used
 - ◆ QueryExpression: actual query
 - Output
 - ◆ Result of Query
 - ◆ SetTerminationTime
 - Request that termination time of this service be changed
 - Input: client timestamp and new termination time
 - Output: service timestamp and current termination time
 - ◆ Destroy
 - Explicit destruction request, returns an ack

Handle Resolver

- Resolves GSH into GSR
 - ◆ Optionally, the client can do it by itself
- Interface
 - ◆ FindByHandle
 - Input: GSH & unsatisfactory GSRs
 - Output: GSR
 - Faults: invalidHandle, no valid references, etc.

Lifecycle

- GS instances created by factory or manually
- Destroyed explicitly or via soft state
 - ◆ Negotiation of initial lifetime with a factory (=service supporting Factory interface)
 - ◆ Lifetime can subsequently be extended by sending “keepalive” messages
- Soft state lifetime management avoids
 - ◆ Explicit client teardown of complex state
 - ◆ Prevents resource “leaks” in hosting environments

GS Creation: Factory

- Creates a new service instance
 - ◆ Reliable once and only once creation
- Interface
 - ◆ CreateService
 - Input:
 - ◆ TerminationTime
 - ◆ ServiceParameters (specific to a service)
 - Output: ServiceTimestamp information & Service Locator
- ServiceLocator can be used to obtain GSH

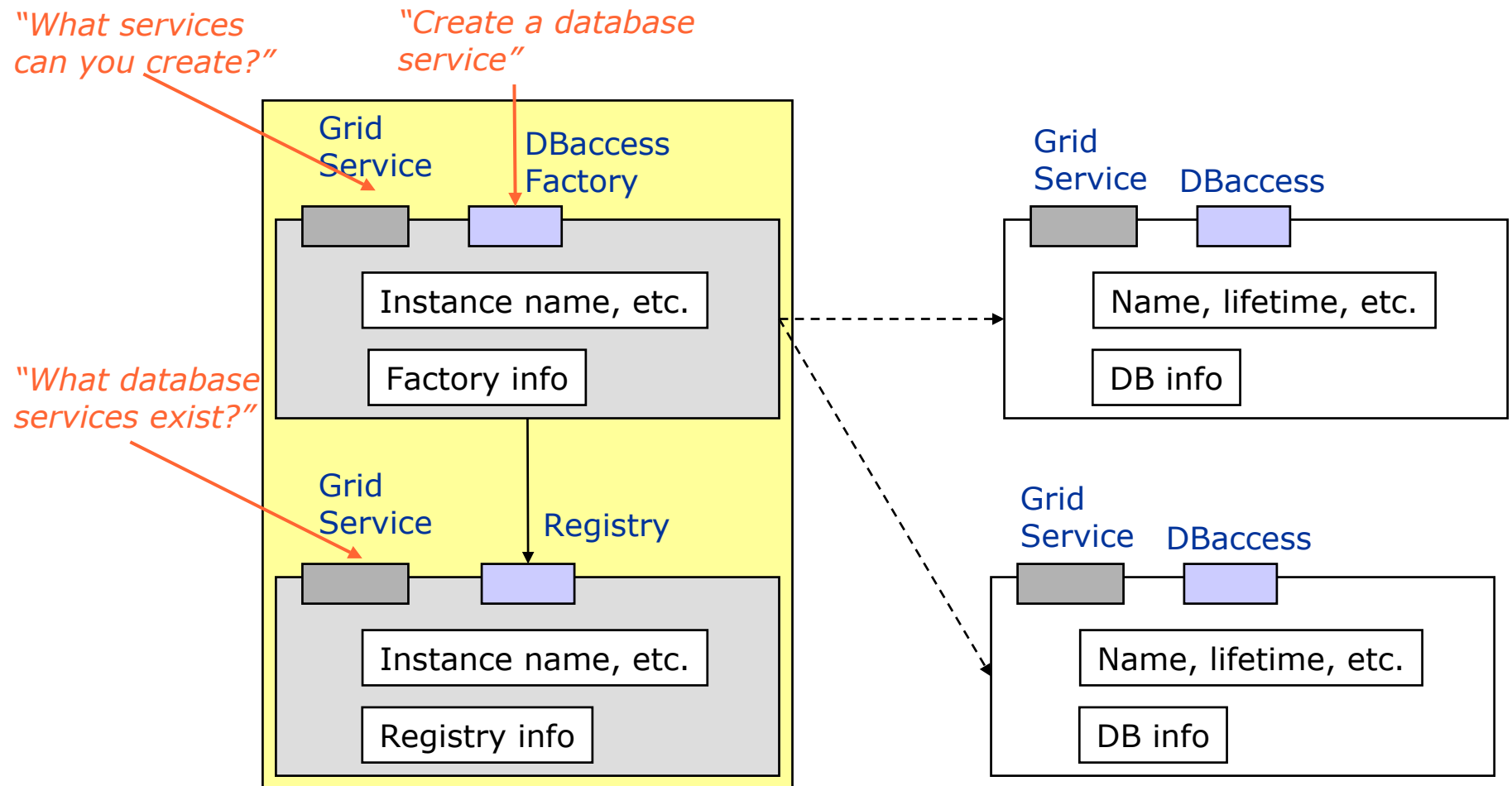
Grid Service Termination

- Explicit destruction
 - ◆ Destroy operation in the Grid Service
- Soft-state destruction
 - ◆ Allowing the termination time to expire
 - ◆ SetTerminationTime operation resets the value of the TerminationTime SDE
 - ◆ Reaffirmation of interest does not guarantee that the service will stay alive

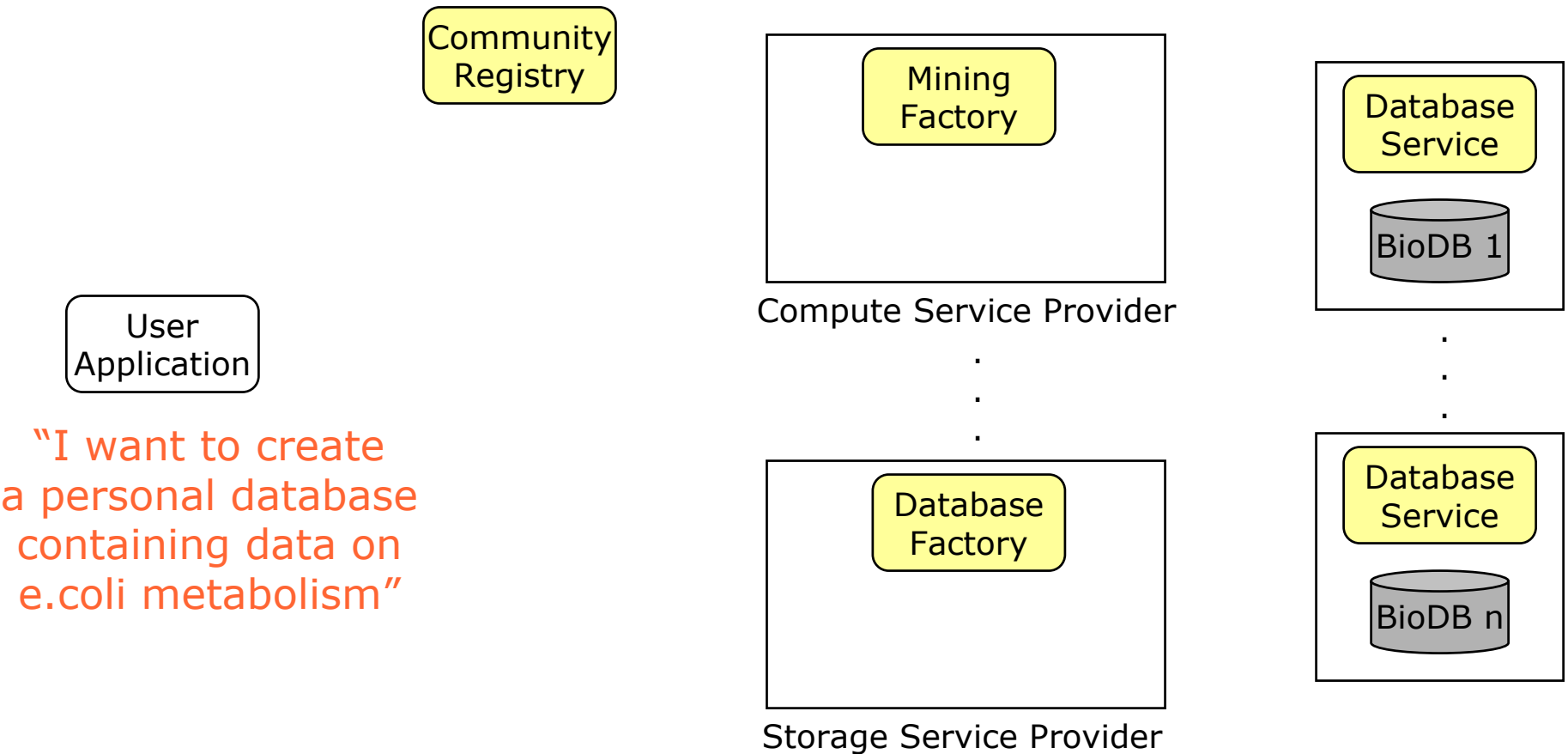
Registry

- The Registry interface may be used to register Grid service instances with a registry
 - ◆ A set of Grid services can periodically register their GSHs into a registry service, to allow for discovery of services in that set
- Registrations maintained in a service data element associated with Registry interface
 - ◆ Standard discovery mechanisms can then be used to discover registered services
 - ◆ Returns a WS-Inspection document containing the GSHs of a set of Grid services

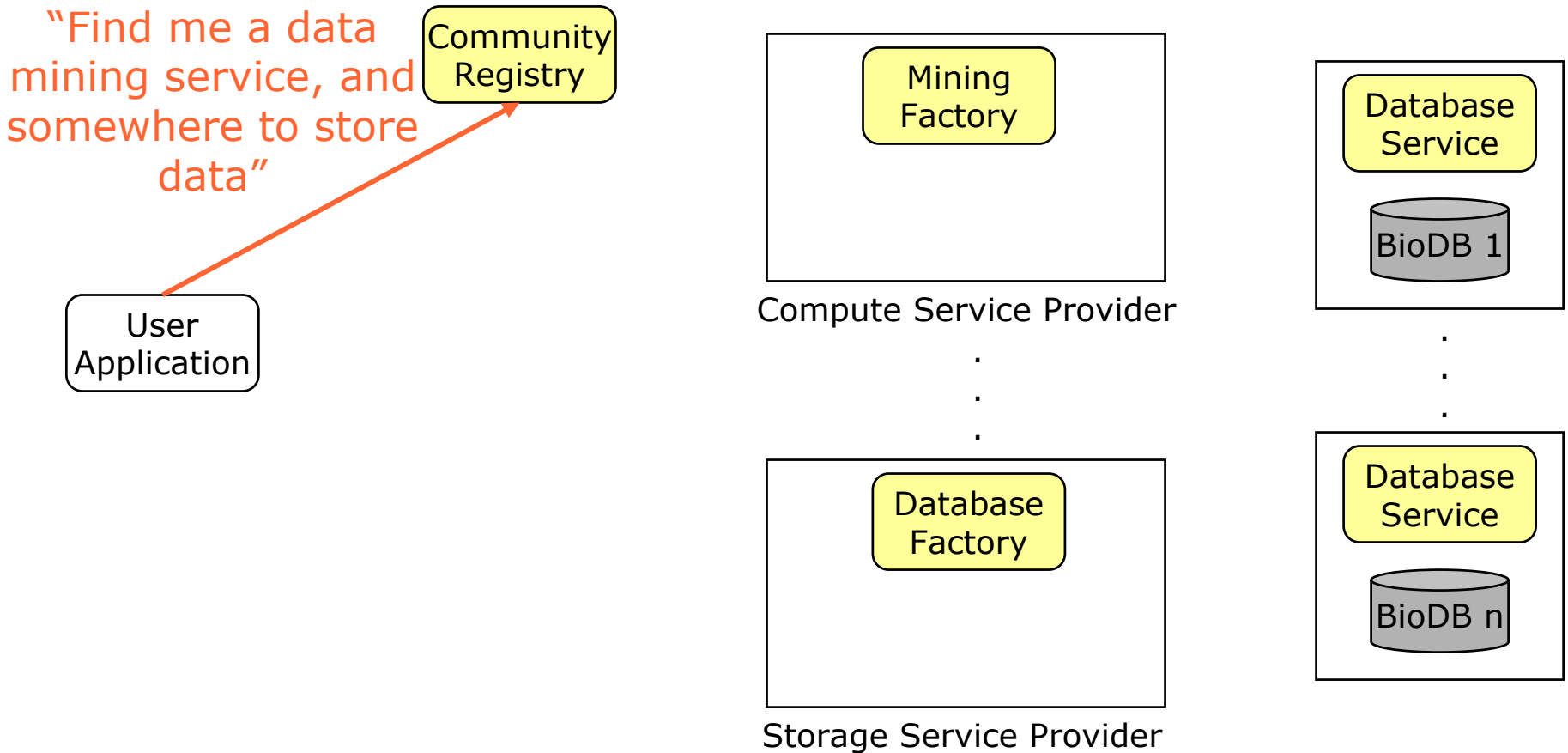
Transient Database Services



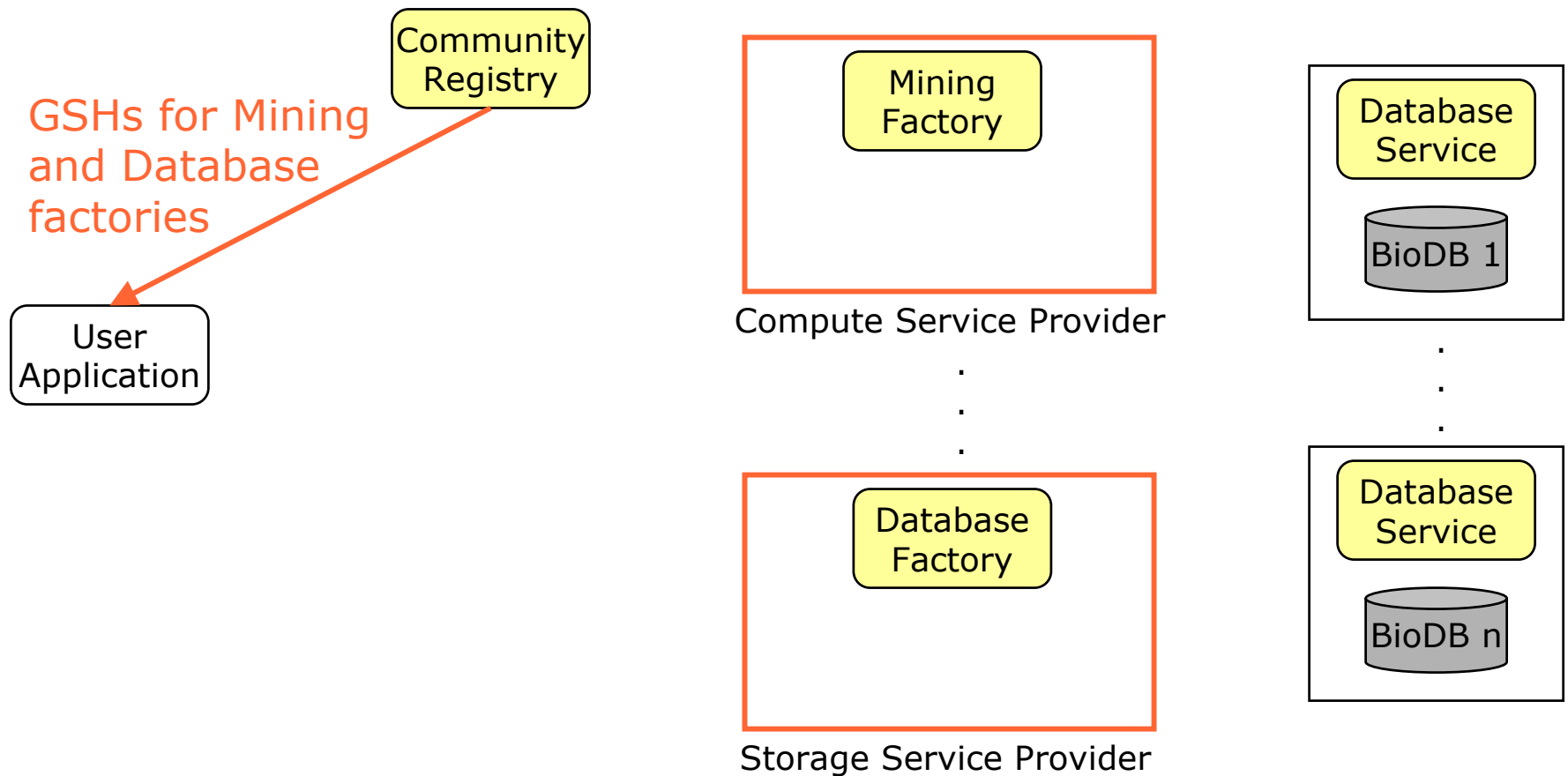
Example: Data Mining for Bioinformatics



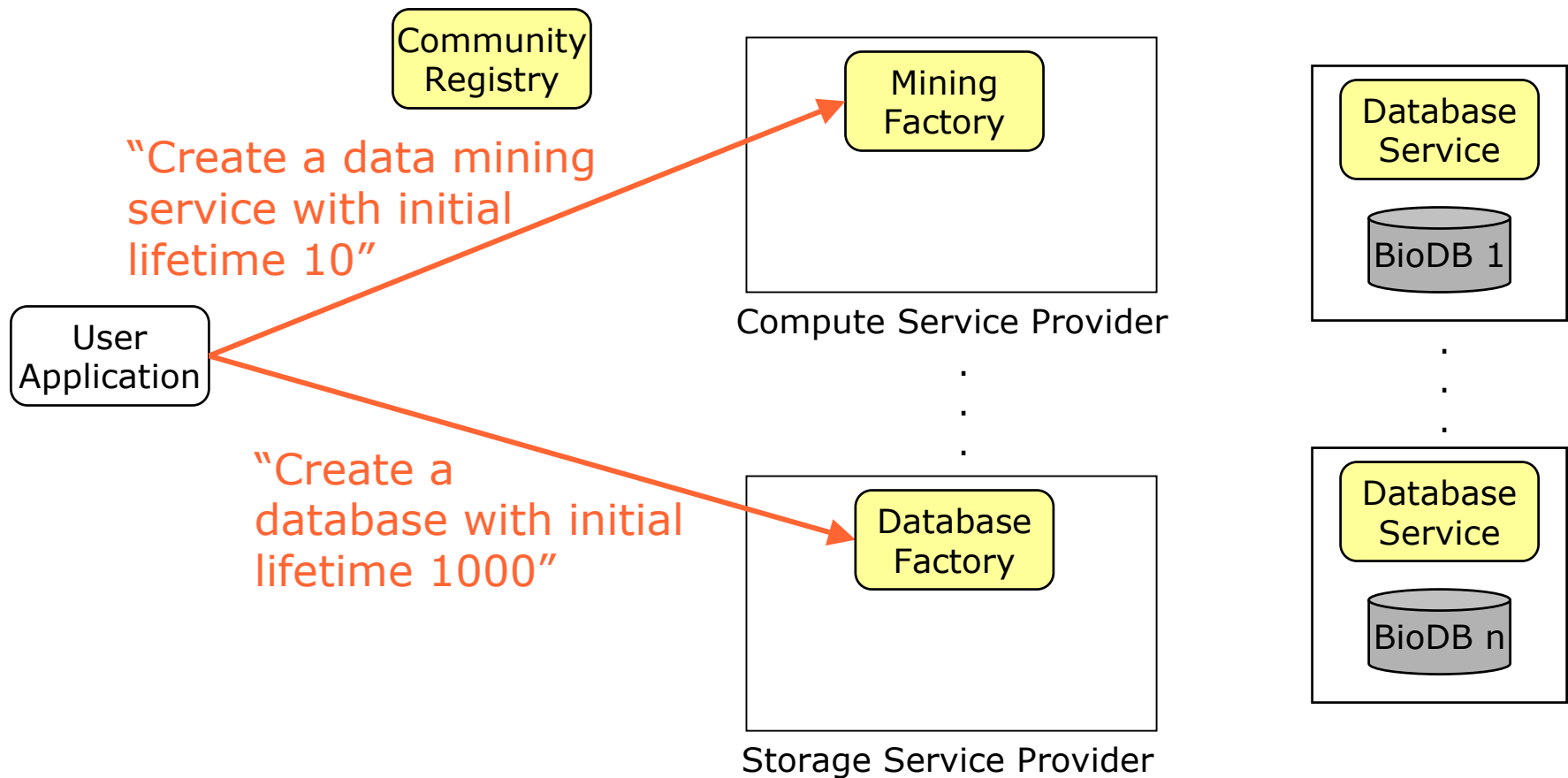
Example: Data Mining for Bioinformatics



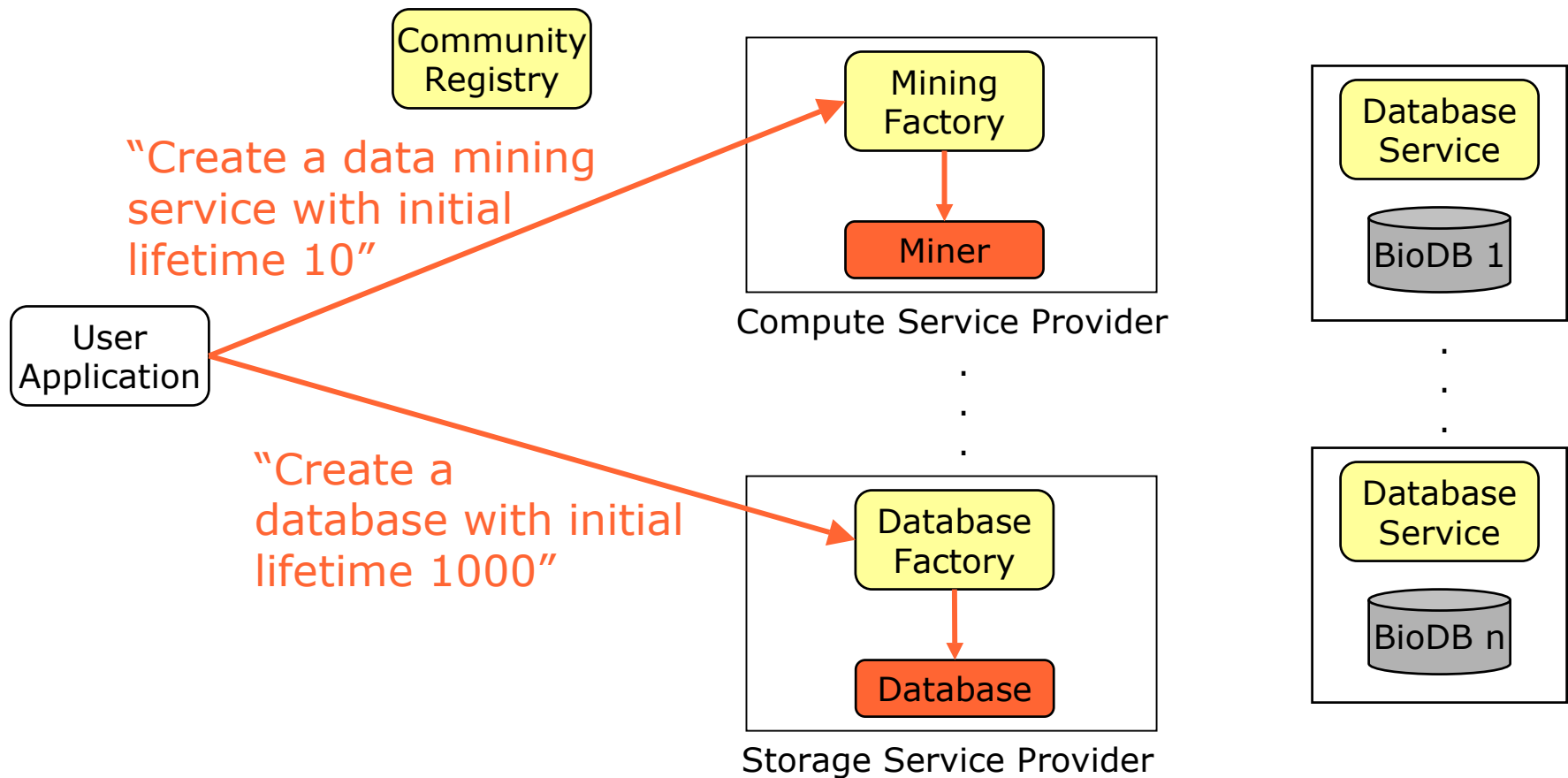
Example: Data Mining for Bioinformatics



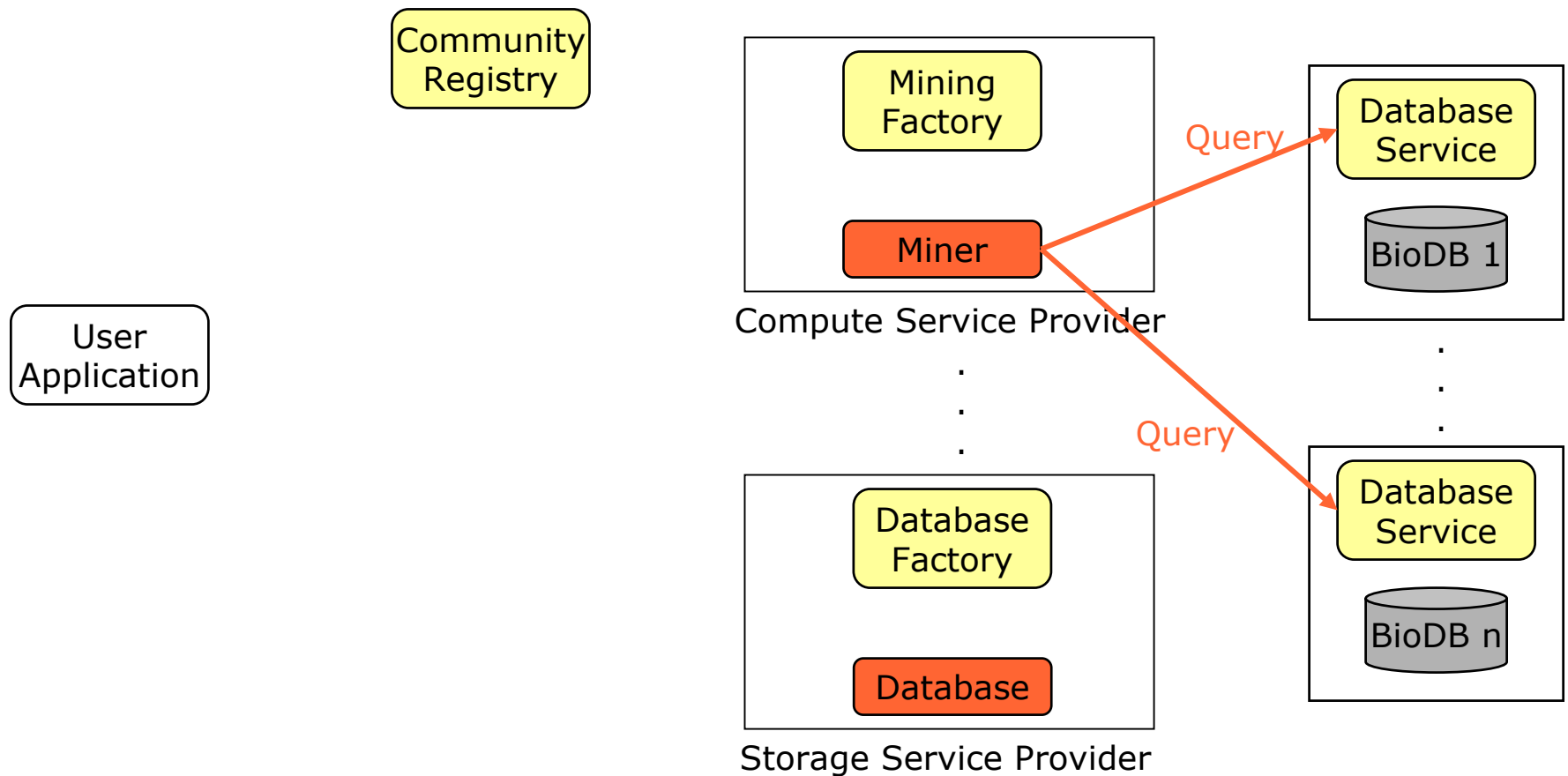
Example: Data Mining for Bioinformatics



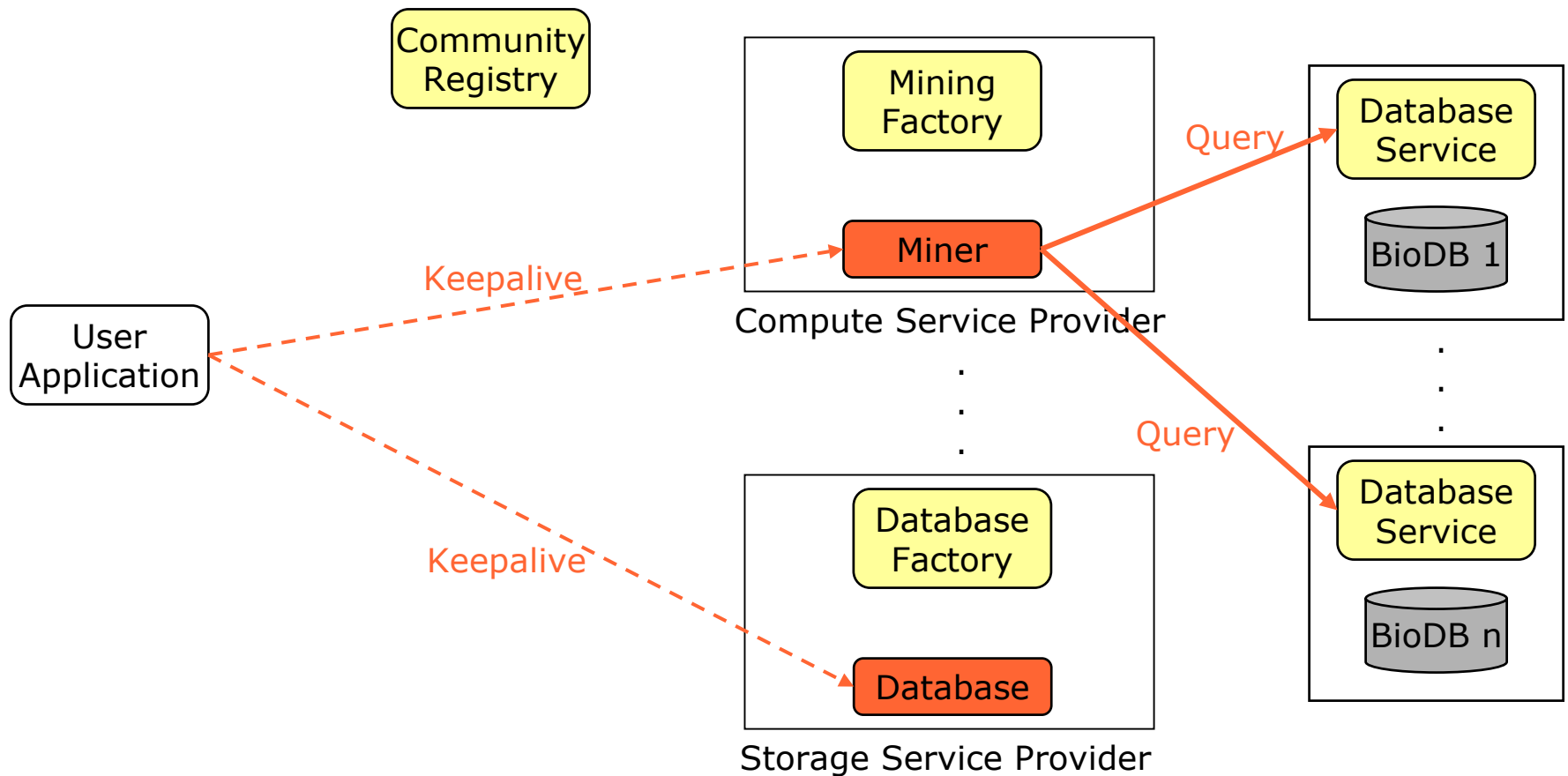
Example: Data Mining for Bioinformatics



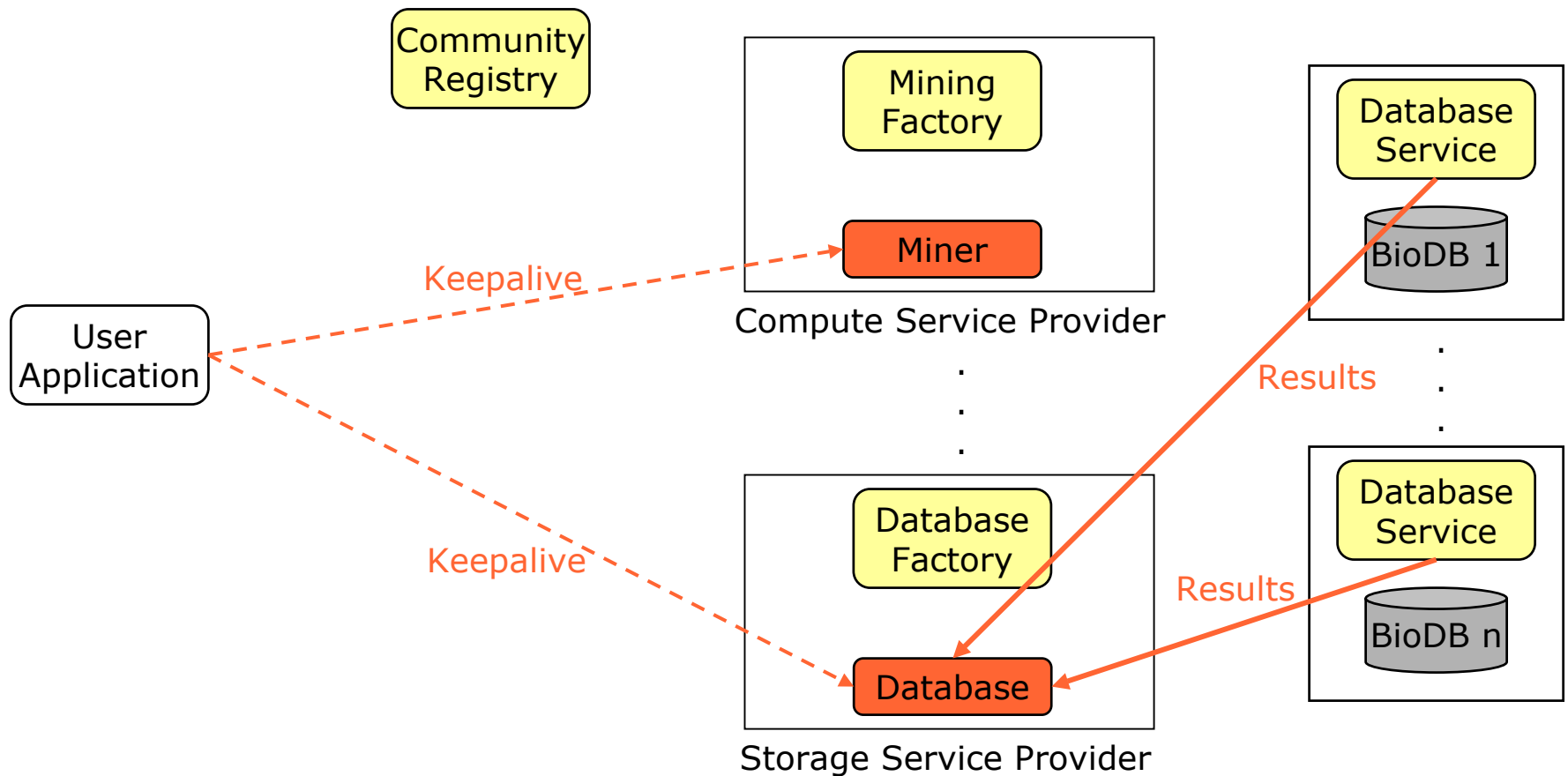
Example: Data Mining for Bioinformatics



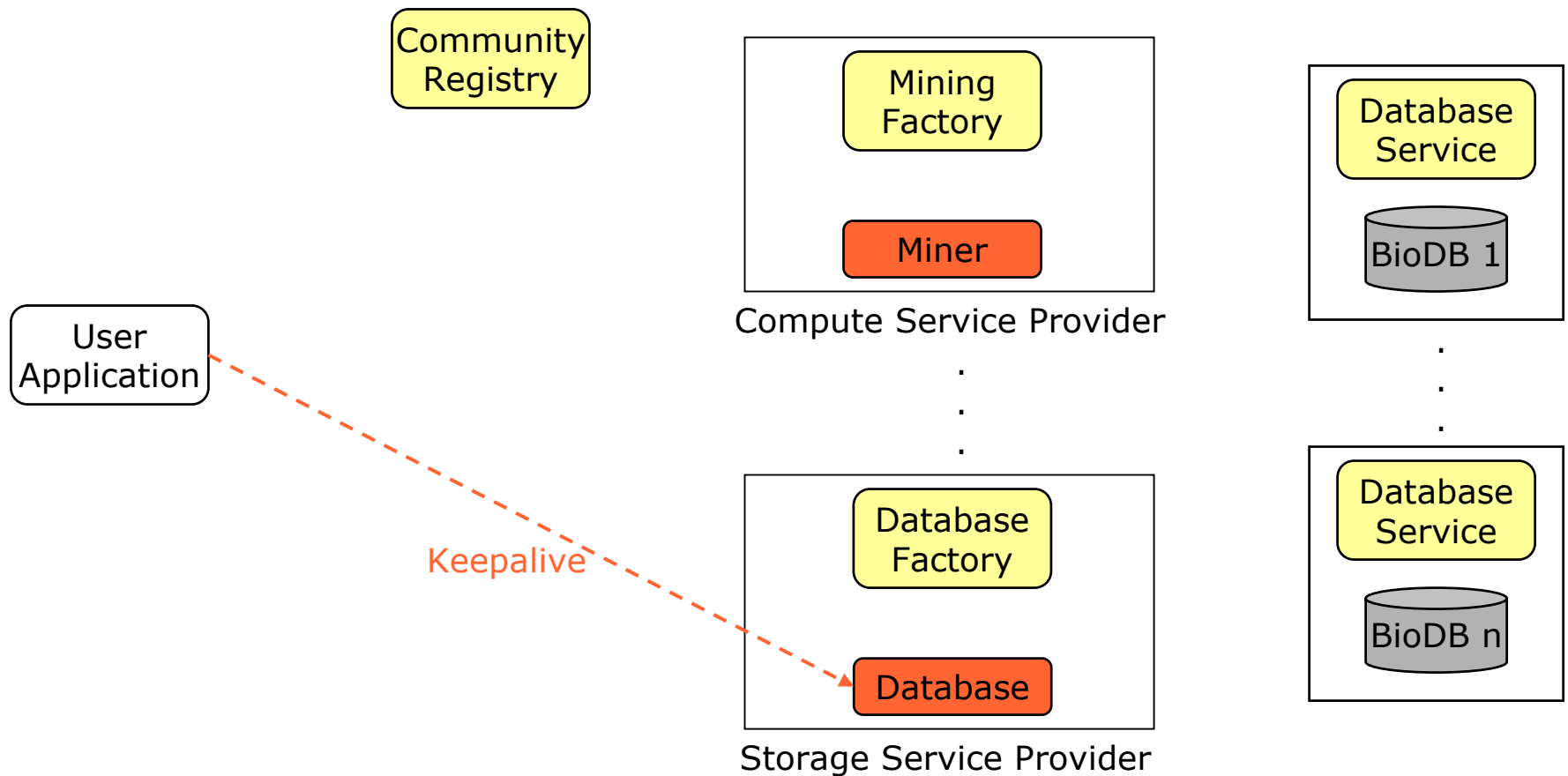
Example: Data Mining for Bioinformatics



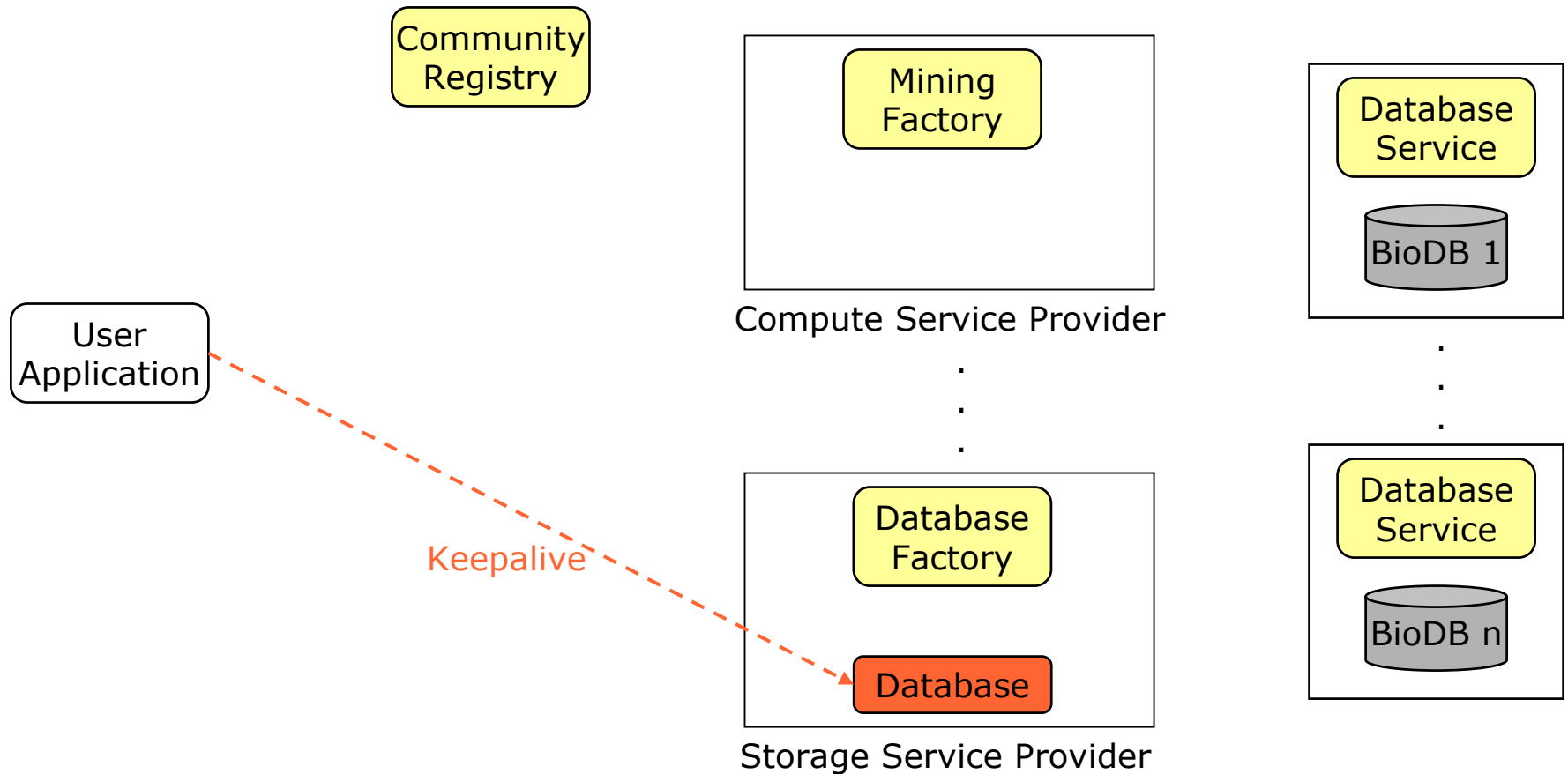
Example: Data Mining for Bioinformatics



Example: Data Mining for Bioinformatics



Example: Data Mining for Bioinformatics

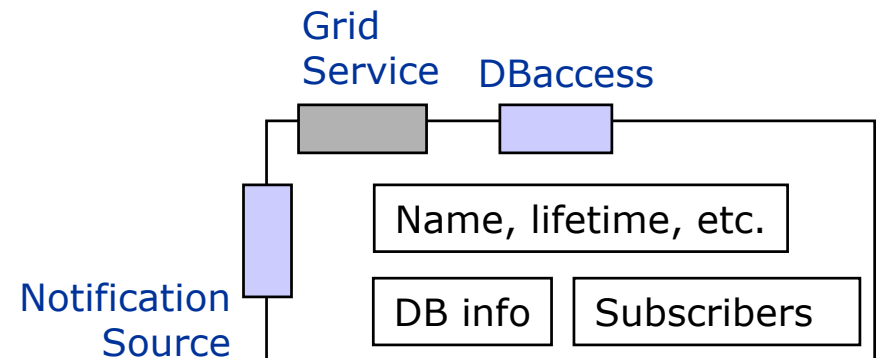
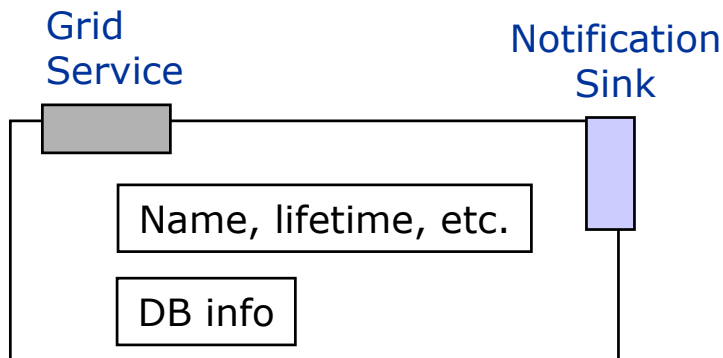


Notification Interfaces

- NotificationSource for client subscription
 - ◆ One or more *notification generators*
 - Generates notification message of a specific type
 - Typed *interest statements*: E.g., Filters, topics, ...
 - Supports messaging services, 3rd party filter services, ...
 - ◆ Soft state subscription to a generator
- NotificationSink for asynchronous delivery of notification messages
- A wide variety of uses are possible
 - ◆ E.g. Dynamic discovery/registry services, monitoring, application error notification, ...

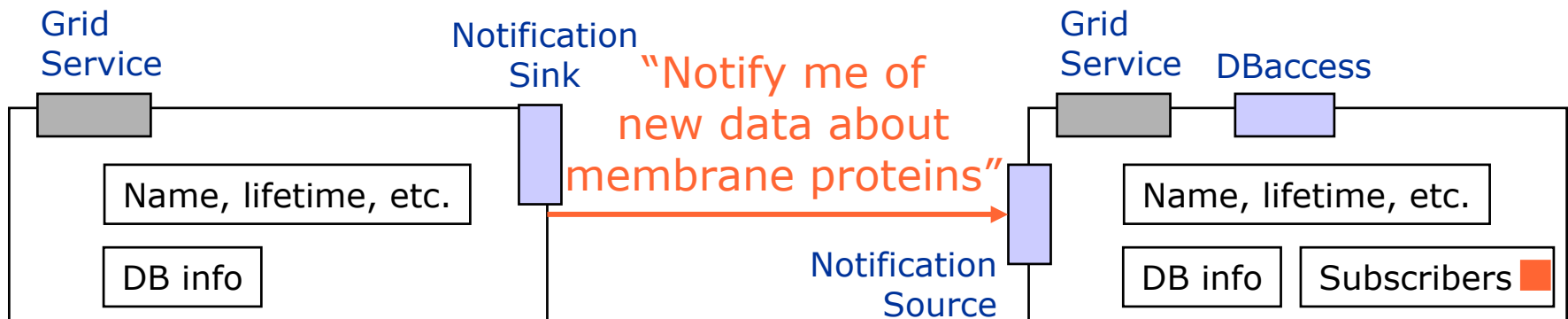
Notification Example

- Notifications can be associated with any (authorized) service data elements



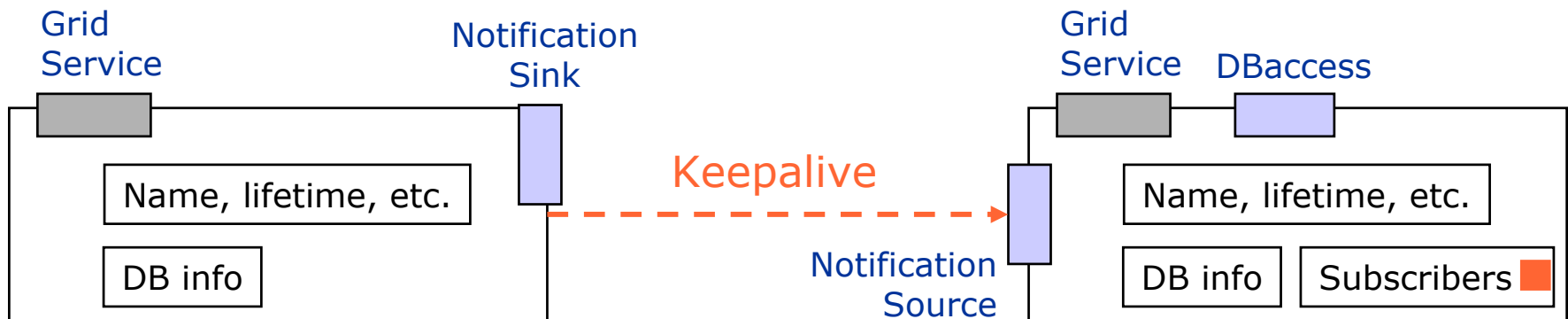
Notification Example

- Notifications can be associated with any (authorized) service data elements



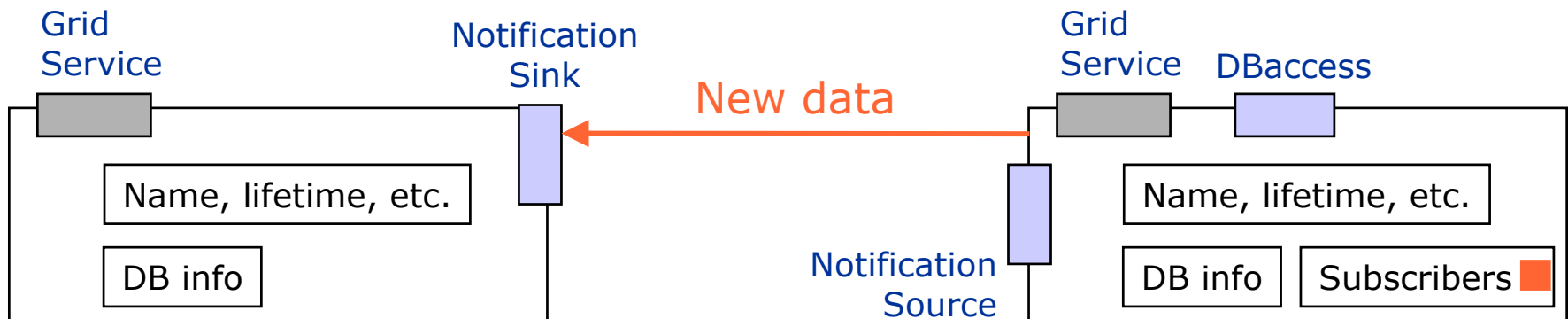
Notification Example

- Notifications can be associated with any (authorized) service data elements



Notification Example

- Notifications can be associated with any (authorized) service data elements



Implementing a Grid Service

- Write WSDL for a service
 - ◆ Association with Grid Service
- Generate stubs and skeletons based on WSDL
 - ◆ WSDL2Java
- Provide implementation of a service
- Implement a factory
 - ◆ Factory WSDL
 - ◆ Generate stubs and skeletons
 - ◆ Provide an implementation
 - ◆ Deploy in .wsdd file
- Provide implementation of a client
- Invoke services

Overview

- Introducing the main players
 - ◆ Grid Computing
 - ◆ Globus Toolkit
 - ◆ Web Service (example)
- The Shape of an OGSA Grid
- The Open Grid Services Architecture
 - ◆ Grid Services: what are they? (example)
 - ◆ WSDL conventions and extensions
 - ◆ OGSA interfaces and behaviors (examples)
- ➔ ◆ **OGSA Security**
- OGSA: status and future

Grid Security Challenges

- Integration Issues
 - ◆ Existing services need to be used
 - ◆ Extensible architecture
- Interoperability Issues
 - ◆ Protocol, policy, and identity level
 - ◆ Quality of Protection (QoP)
- Trust Issues
 - ◆ Definition, management and enforcement of trust

Grid Security Requirements

- Authentication
- Delegation
- Single sign-on
- Credential Lifespan and renewal
- Authorization
- Privacy
- Confidentiality
- Integrity
- Policy exchange
- Secure logging
- Assurance
- Manageability
- Firewall traversal
- Securing the OGSA infrastructure...

Grid Security in OGSA

- Two documents
 - ◆ OGSA Security Roadmap defines a set of required services and indicates for each if
 - Is provided by WS Security specs
 - May be provided by WS Security specs
 - Requires standardized profile/mechanisms and/or extensions for WS Security specs
 - ◆ The Security Architecture for Open Grid Services
 - ◆ Available at www.globus.org/ogsa/security
- GGF working group

Overview

- Introducing the main players
 - ◆ Grid Computing
 - ◆ Globus Toolkit
 - ◆ Web Service (example)
- The Shape of an OGSA Grid
- The Open Grid Services Architecture
 - ◆ Grid Services: what are they? (example)
 - ◆ WSDL conventions and extensions
 - ◆ OGSA interfaces and behaviors (examples)
 - ◆ OGSA Security
- ➔ • OGSA: status and future

OGSA and the Globus Toolkit

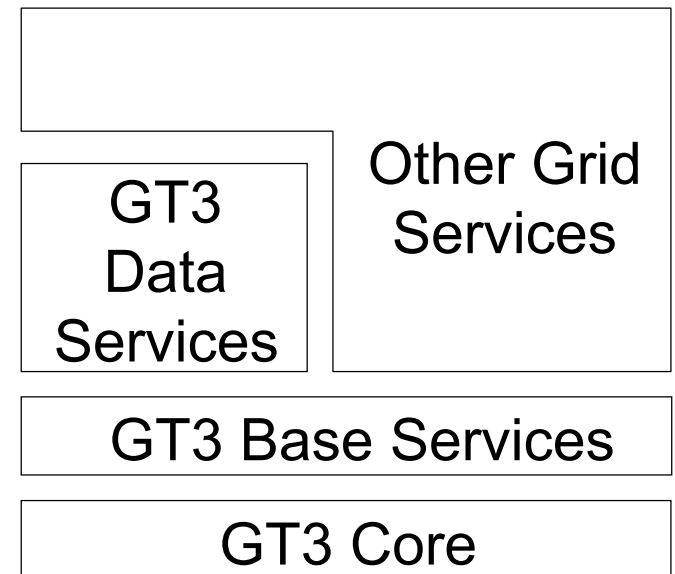
- Technically, OGSA enables
 - ◆ Refactoring of protocols (GRAM, MDS-2, etc.)—*while preserving all GT concepts/features!*
 - ◆ Integration with hosting environments: simplifying components, distribution, etc.
 - ◆ Greatly expanded standard service set
- Pragmatically, we are proceeding as follows
 - ◆ Develop open source OGSA implementation
 - Globus Toolkit 3.0; supports Globus Toolkit 2.0 APIs
 - ◆ Partnerships for service development
 - ◆ Also expect commercial value-adds

GT3: an OGSA-Compliant Globus Toolkit

- Open source implementation of OGSA from the Globus Project
- Globus Toolkit 3.0 (GT3)
 - ◆ first prototype Grid service implementation demonstrated on January 29, 2002
 - ◆ Several OGSI Technology Preview releases throughout the year
 - ◆ Alpha release expected end of 2002
 - ◆ For details see www.globus.org/ogsa
- Also, other implementations
 - ◆ Unicore, LBNL...

GT3 Structure

- GT3 Core
 - ◆ Implements Grid service interfaces & behaviors
 - ◆ Reference impln of evolving standard
 - ◆ Java, C, Python, C++...
- GT3 Base Services
 - ◆ Evolution of current Globus Toolkit capabilities
 - ◆ Backward compatible
- Many other Grid services



GT2 vs GT3 Strategy

- GT3 lets you do all the things you can do with GT2
 - ◆ Same familiar services: GRAM, GridFTP, etc.
 - ◆ Strong commitment to providing compatibility APIs
 - ◆ We do not enforce any particular programming model
- But GT3 also allows you to do many other things
 - ◆ Service orientation
 - ◆ Virtualization opportunities
 - ◆ New capabilities

Community Involvement: GGF

- GGF Working Groups:
 - ◆ OGSI-WG
 - refinement of the infrastructure-related portions of OGSA.
 - Formed February 2002
 - Led by S. Tuecke, D. Snelling
 - ◆ OGSA-WG
 - Architectural aspects
 - Formed July 2002, led by I. Foster, J. Nick, D. Gannon
 - ◆ OGSA Security WG
 - Formed July 2002, led by F. Siebenlist, N. Nagaratnam
 - ◆ Proposed: Java binding



Grids and OGSA: Research Challenges

- Grids pose profound problems, e.g.
 - ◆ Management of virtual organizations
 - ◆ Delivery of multiple qualities of service
 - ◆ Autonomic management of infrastructure
 - ◆ Software and system evolution
- OGSA provides foundation for tackling these problems in a rigorous fashion?
 - ◆ Structured establishment/maintenance of global properties
 - ◆ Reasoning about total system properties

Summary:

Evolution of Grid Technologies

- Initial exploration (1996-1999; Globus 1.0)
 - ◆ Extensive appln experiments; core protocols
- Data Grids (1999-??; Globus 2.0+)
 - ◆ Large-scale data management and analysis
- Open Grid Services Architecture (2001-??, Globus 3.0)
 - ◆ Integration w/ Web services, hosting environments, resource virtualization
 - ◆ Databases, higher-level services
- Radically scalable systems (2003-??)
 - ◆ Sensors, wireless, ubiquitous computing

Summary

- The Grid problem: Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations
- Grid architecture: Protocol, service definition for interoperability & resource sharing
- Globus Toolkit a source of protocol and API definitions—and reference implementations
 - ◆ And many projects applying Grid concepts (& Globus technologies) to important problems
- Open Grid Services Architecture represents (we hope!) next step in evolution

Bibliography

- **Grids and the Globus Toolkit**

- ◆ General information: www.globus.org
- ◆ Global Grid Forum: www.gridforum.org
- ◆ Technical Papers: <http://www.globus.org/research/papers.html>
- ◆ The Grid: Blueprint for a New Computing Infrastructure, I. Foster, C. Kesselman, Morgan-Kaufmann, 1999

- **Web Services**

- ◆ XML Schema Part 0: Primer, W3C Recommendation, at www.w3.org/TR/xmlschema-0/
- ◆ Web Services Description Language (WSDL) Version 1.2, W3C Working Draft 9 July 2002, at <http://www.w3.org/TR/2002/WD-wsdl12-20020709/>
- ◆ Building Web Services with Java, S. Graham, S. Simeonov, T. Boubez, D. Davis, G. Daniels, Y. Nakamura, R. Neyama, Sams Publishing, December 2001
- ◆ Web Services Essentials, E. Cerami, O'Reilly, January 2002

- **Grid Services**

- ◆ General information at www.globus.org/ogsa
- ◆ GGF OGSI-WG at <http://www.gridforum.org/ogsi-wg/>
- ◆ The Physiology of the Grid, I. Foster, C. Kesselman, J. M. Nick, S. Tuecke, at www.globus.org/ogsa
- ◆ Grid Services Specification, S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, at www.globus.org/ogsa
- ◆ The Security Architecture for Open Grid Services, N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, S. Tuecke at www.globus.org/ogsa/Security
- ◆ OGSA Security Roadmap, F. Siebenlist, V. Welch, S. Tuecke, I. Foster, N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, at www.globus.org/ogsa/Security

Acknowledgements

- Grid computing, Globus Project, and OGSA
 - ◆ Ian Foster, Steve Tuecke @ANL
 - ◆ Carl Kesselman @ USC/ISI
 - ◆ Talented team of scientists and engineers at ANL, USC/ISI, elsewhere (see www.globus.org)
- Open Grid Services Architecture (OGSA)
 - ◆ Karl Czajkowski @ USC/ISI, Jeff Nick, Steve Graham, Jeff Frey @ IBM, www.globus.org/ogsa
- Grid security, OGSA Security
 - ◆ Frank Siebenlist, Von Welch
- Support from DOE, NASA, NSF, IBM, Microsoft